Implementation of isogeny-based cryptography

2/23/2021

NIST PQC Seminar Series

David Jao, University of Waterloo and evolution Q, Inc.
Isogenies

An isogeny is a non-constant morphism between elliptic curves which preserves base points.

For computational purposes, the emphasis is on separable isogenies over finite fields.
Let $\phi : E_1 \to E_2$ be an isogeny.

Define $\ker \phi = \{ P \in E_1 \mid \phi(P) = O_E \}.$

Obviously, $\phi$ determines $\ker \phi.$

**Theorem:** If $\phi$ is separable, then $\ker \phi$ determines $\phi$ up to isomorphism, and

$$\deg \phi = | \ker \phi |$$
Vélu's formulas (1971):

Let \( S \subseteq E \) be a finite subgroup. Then

\[
\phi_x(P) \overset{\text{def}}{=} x(P) + \sum_{Q \in S \setminus \{O_E\}} [x(P+Q) - x(Q)]
\]

\[
\phi_y(P) \overset{\text{def}}{=} y(P) + \sum_{Q \in S \setminus \{O_E\}} [y(P+Q) - y(Q)]
\]

defines a separable isogeny \( \phi(P) = (\phi_x(P), \phi_y(P)) \) with \( \ker \phi = S \).
Vélu's formulas require $O(d)$ operations ($d = \deg \phi$). However, these operations take place in a field of definition for the points in $\ker \phi$.

- It is possible for an isogeny $\phi$ to be defined over $F$ (equivalently, for $\ker \phi$ to be defined over $F$) even when not all points in $\ker \phi$ are defined over $F$.

- If extensions of a finite field are used, then Vélu's formulas cost $\approx O(d^n)$ basic operations

- Vélu (Bernstein et al. in:cr/2020/341) is faster for large $d$
Conveignes Rostovster Stolbunov scheme (CRS)
(Deviced in 1996, published 2006)

\[ E \xrightarrow{\phi_\alpha} \alpha E \]
\[ \phi_{\beta} \downarrow \quad \downarrow \]
\[ \beta E \rightarrow \alpha \beta E = \beta \alpha E \]

\[ E/F_p \quad \text{ordinary} \]
\[ \alpha, \beta \in \text{Cl} (\text{End} (E)) \]
\[ * \quad \text{is the complex multiplication operator, defined by} \]
\[ \ker \phi_\alpha = \bigwedge_{\psi \in \alpha} \ker \psi. \]

- For this scheme to be feasible, \( \alpha \) and \( \beta \)
must be chosen to be smooth ideals.
- The scheme is still very slow because the kernel points
  are defined over a field extension.
Supersingular Isogeny Diffie–Hellman (SIDH) (Jao, De Feo, and Plût, 2011)

Uses an assumption formulated by Charles, Goren, and Lauter (2006)

\[
E \xrightarrow{\phi_A} E_A \quad E/\mathbb{F}_p \quad \text{supersingular, } p = 2^e 3^f - 1,
\]

\[
\phi_B \quad \text{both defined over } \mathbb{F}_{p^2}
\]

\[
E_B \xrightarrow{\phi_B} E_{AB}
\]

Alice sends \((E_A, \phi_A | E[3^f])\)

Bob sends \((E_B, \phi_B | E[2^e])\)

\[
\ker \phi_A \subset E[2^e], \deg \phi_A = 2^e
\]

\[
\ker \phi_B \subset E[3^f], \deg \phi_B = 3^f
\]

- Since all points are defined over \(\mathbb{F}_{p^2}\), "fast" Vélu can be used. Furthermore, only isogenies of degree 2 or 3 are needed.
Commutative SIDH (CSIDH)
(Castryck, Lange, Martindale, Panny, Renes, 2018)

\[ E \rightarrow \alpha \cdot E \]
\[ \downarrow \quad \downarrow \]
\[ b \cdot E \rightarrow \alpha \cdot b \cdot E \]

\[ \frac{p}{\prod_{i=1}^{n} \ell_i} - 1, \quad \text{tr} \,(E) = 0 \]

- Construction of \( E \) and \( p \) allows the use of isogenies of degree \( \ell_i \) with all kernel points defined over \( \mathbb{F}_{p^2} \).
- Hence "fast" Vélu can be used, so CSIDH is faster than CRS.
- CSIDH is slower than SIDH (because CSIDH uses degrees \( \ell_i \geq 3 \) and SIDH uses degree = 2 or 3)
Security of SIDH / SIKE

SIDH can be broken by an invalid key attack
- Galbraith, Petit, Shani, Ti, ia.cr/2016/859

SIKE = SIDH + HHK / FO

Proposed parameter sizes:

<table>
<thead>
<tr>
<th>log₂ p</th>
<th>1PKI (bytes)</th>
<th>NIST level</th>
</tr>
</thead>
<tbody>
<tr>
<td>434</td>
<td>330</td>
<td>1</td>
</tr>
<tr>
<td>503</td>
<td>378</td>
<td>2</td>
</tr>
<tr>
<td>610</td>
<td>462</td>
<td>3</td>
</tr>
<tr>
<td>751</td>
<td>564</td>
<td>5</td>
</tr>
</tbody>
</table>

Security analysis:
- Longa, Wang, Saefer, ia.cr/2020/1457
- Jaques & Schrottenloher, ia.cr/2020/424
- Jaques & Schanck, ia.cr/2019/103
protocol
iso-geny operations
curve arithmetic
$F_p^2$ arithmetic
$\text{mod } p$ arithmetic
integer arithmetic
hardware operations
Key compression
Optimal strategies
Vélu sqrt
3-point Montgomery ladder
Karatsuba multiplication and squaring
Montgomery modular multiplication
specialized instructions
custom hardware
Optimal strategies

Let \( \ker \phi = \langle p \rangle \), \( \text{ord}(p) = 2^e \)
\( = \deg \phi \).

Write \( \phi = \phi_1 \circ \phi_2 \circ \ldots \circ \phi_e \)
where \( \deg \phi_i = 2 \), \( i = 1 \) to \( e \).

Then:
\( \ker \phi_1 = \langle 2^{e-1} p \rangle \)
\( \ker \phi_2 = \langle \phi_1(2^{e-2} p) \rangle \)
\( = \langle 2^{e-2} \phi_1(p) \rangle \)

etc.

Green edges denote optimal subgraph satisfying all required dependencies

For CSIDH: Hutchinson et al., in: cr/2019/1121
Protocol optimizations - Point compression for SIDH

In SIDH, Alice sends \((E_A, \phi_A|_{E[3^f]})\) to Bob.

- Naive method: Choose basis \(\{P, Q\}\) for \(E[3^f]\),
  send coefficients of \(E_A\) and \(\phi_A(P), \phi_A(Q)\) to Bob.

- Compressed representation (saves 50% key size):
  i) Fix a basis \(\{PA, QA\}\) for \(E_A[3^f]\).
  ii) Send the coefficient matrix of \(\{\phi_A(P), \phi_A(Q)\}\)
      with respect to \(\{PA, QA\}\).
  iii) Send an isomorphism invariant of \(E_A\) instead of \(E_A\).

Latest results: Pereira et al. ia.cr/2020/43

- Code available at [https://github.com/microsoft/PQCrypto-SIDH](https://github.com/microsoft/PQCrypto-SIDH)
- 21% faster encryption, 76% faster decryption
  compared to previous compressed SIDH.
Curve arithmetic

Montgomery curves \((By^2 = x^3 + Ax^2 + x)\) are the current efficiency leaders, but other models have been studied:

- Huff (ia.cr/2020/526)
- Hessian (ia.cr/2019/1480)
- Edwards (ia.cr/2019/843, ia.cr/2019/110)
- Edwards + Huff (ia.cr/2011/430)
$\mathbb{F}_p^2$ arithmetic

$\mathbb{F}_p^2 = \mathbb{F}_p[i], \ i = \sqrt{-1}$

$(a+bi)(c+di) = (ac - bd) + (ad + bc)i$

$= (ac - bd) + [(a+b)(c+d) - ac - bd]i$

$(a+bi)^2 = (a^2 - b^2) + 2abi = (a+b)(a-b) + 2ab i$

Mod $p$ arithmetic

Montgomery representation
- Karatsuba
- Squaring

Polynomial representation (Bouvier & Imbert ia.cr/2020/1385)
Specialized instructions

Example: UMAAL on ARM
\[ a, b, c, d \rightarrow (c, d \leftarrow a \cdot b + c + d) \]

<table>
<thead>
<tr>
<th>a b c d</th>
<th>a \cdot b + c + d</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 6 0 0</td>
<td>54</td>
</tr>
<tr>
<td>8 6 0 0</td>
<td>48</td>
</tr>
<tr>
<td>9 5 5 8</td>
<td>58</td>
</tr>
<tr>
<td>8 5 4 5</td>
<td>49</td>
</tr>
<tr>
<td>9 4 0 0</td>
<td>36</td>
</tr>
<tr>
<td>7 6 6 9</td>
<td>57</td>
</tr>
<tr>
<td>8 4 3 4</td>
<td>39</td>
</tr>
<tr>
<td>7 5 9 5</td>
<td>49</td>
</tr>
<tr>
<td>7 4 3 4</td>
<td>35</td>
</tr>
</tbody>
</table>
ARM Cortex M4 implementation

Anustasova, Azavdevakhsh, Mozaffari-Kermani (in.cn/2021/115)

<table>
<thead>
<tr>
<th>P (bits)</th>
<th>434</th>
<th>503</th>
<th>610</th>
<th>751</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (ms)</td>
<td>850</td>
<td>1200</td>
<td>2430</td>
<td>3690</td>
</tr>
<tr>
<td>% improvement over SIKE rd.3</td>
<td>22.5%</td>
<td>21.6%</td>
<td>17.5%</td>
<td>19.5%</td>
</tr>
</tbody>
</table>

Key ideas:
- interleave additions/subtractions
- reduce mod \( p+1 \) instead of mod \( p \)
- use floating point registers as faster memory
<table>
<thead>
<tr>
<th>Year</th>
<th>H</th>
<th>M</th>
<th>S</th>
<th>X</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1966</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2014</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>1984</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Notes:**
- [2] Elehaut et al. 14th CC 2020

**Dedicated Hardware and HW/SW Co-design**